Requested Patent: EP1202159A2

Title:

METHOD AND SYSTEM UTILIZING DATA FRAGMENTS FOR EFFICIENTLY IMPORTING/EXPORTING REMOVABLE STORAGE VOLUMES ;

Abstracted Patent: EP1202159 ;

Publication Date: 2002-05-02 ;

Inventor(s):

CARLSON WAYNE CHARLES (GB); KISHI GREGORY TAD (GB); PEAKE JONATHAN WAYNE (GB) ;

Applicant(s): IBM (US) ;

Application Number: EP20010308883 20011019 ;

Priority Number(s): US20000694750 20001023 ;

IPC Classification: G06F3/06 ;

Equivalents: JP2002182952 ;

ABSTRACT:

A method and system utilizing data fragments for efficiently importing/exporting a removable storage volume having a number of data files from a first virtual storage system to a second virtual storage system. The method includes writing data fragments to the end of the removable storage volume in the first virtual storage system. The data fragments contain information, such as data file headers, that uniquely identifies the data files residing in the removable storage volume. Next, the removable storage volume is transferred to the second virtual storage system. Upon receipt of the volume, the second virtual storage system updates a tape volume cache in the second virtual storage system utilizing the information contained in the data fragments without having to read each of the data files. In one embodiment, the data fragments include at least one data file, a file header preceding the data file and a data fragment trailer.

(19) Europäis hes Pat ntamt
European Pat nt Office
Offic uropéen des br vets

(11) EP 1 202 159 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
02.05.2002 Bulletin 2002/18

(51) Int Cl.[7]: G06F 3/06

(21) Application number: 01308883.6

(22) Date of filing: 19.10.2001

(84) Designated Contracting States:
AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE TR
Designated Extension States:
AL LT LV MK RO SI

(30) Priority: 23.10.2000 US 694750

(71) Applicant: International Business Machines Corporation
Armonk, NY 10504 (US)

(72) Inventors:
- Carlson, Wayne Charles,
IBM United Kingdom Ltd.
Winchester, Hampshire SO21 2JN (GB)
- Kishi, Gregory Tad, IBM United Kingdom Ltd,
Winchester, Hampshire SO21 2JN (GB)
- Peake, Jonathan Wayne,
IBM United Kingdom Ltd.
Winchester, Hampshire SO21 2JN (GB)

(74) Representative: Burt, Roger James, Dr,
IBM United Kingdom Limited
Intellectual Property Department
Hursley Park
Winchester Hampshire SO21 2JN (GB)

(54) Method and system utilizing data fragments for efficiently importing/exporting removable storage volumes

(57) A method and system utilizing data fragments for efficiently importing/exporting a removable storage volume having a number of data files from a first virtual storage system to a second virtual storage system. The method includes writing data fragments to the end of the removable storage volume in the first virtual storage system. The data fragments contain information, such as data file headers, that uniquely identifies the data files residing in the removable storage volume. Next, the removable storage volume is transferred to the second virtual storage system. Upon receipt of the volume, the second virtual storage system updates a tape volume cache in the second virtual storage system utilizing the information contained in the data fragments without having to read each of the data files. In one embodiment, the data fragments include at least one data file, a file header preceding the data file and a data fragment trailer.

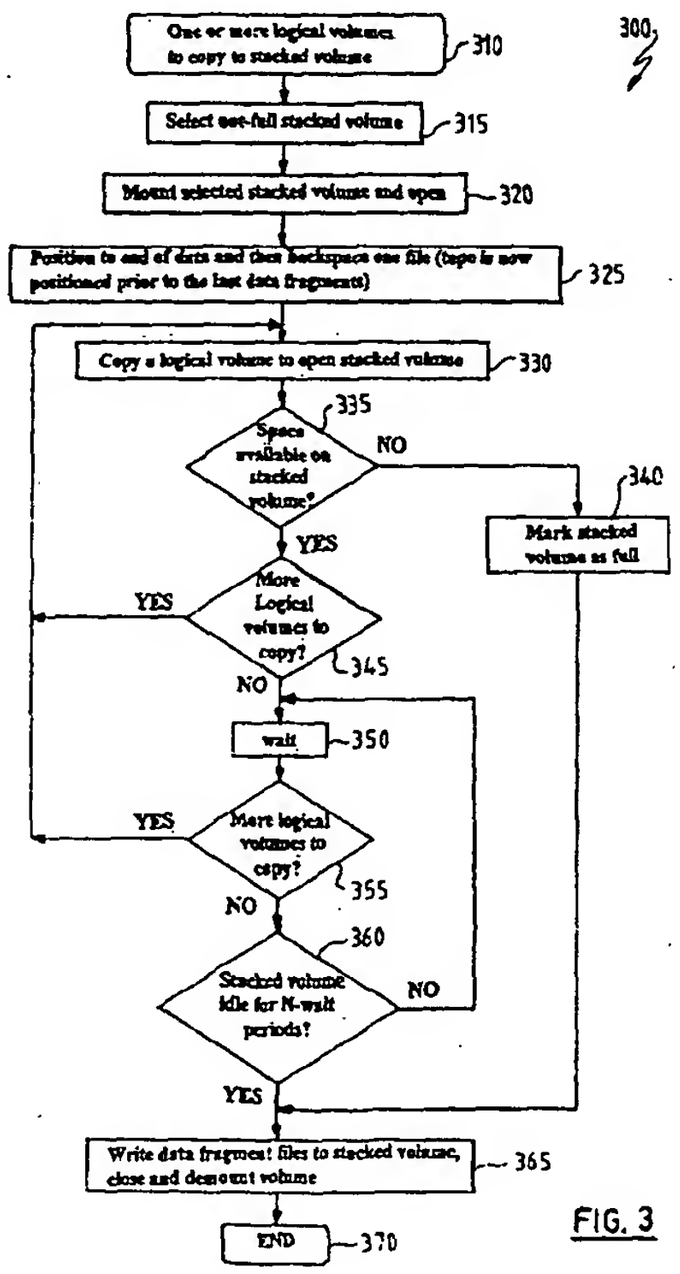

FIG. 3

EP 1 202 159 A2

Descripti n

[0001] The present invention relates in general to data storage and processing and, in particular to virtual storage systems. More particularly, the present invention relates to a method and system utilizing data fragments for efficiently importing/exporting removable storage volumes between virtual storage systems.

[0002] In hierarchical virtual storage systems, intensively used and frequently accessed data is stored in fast but expensive memory. On example of a fast memory is a direct access storage device (DASD). In contrast, less frequently accessed data is stored in less expensive but slower memory. Examples of slower memory are tape drives and disk drive arrays. The goal of the hierarchy is to obtain moderately priced, high-capacity storage while maintaining high-speed access to the stored information.

[0003] One such hierarchical storage system is a virtual tape storage system (VTS), including a host data interface, a DASD, and a number of tape devices. When the host writes a logical volume, or a file, to the VTS, the data is stored as a file on the DASD. Although the DASD provides quick access to his data, it will eventually reach full capacity and a backup or secondary storage system will be needed. An International Business Machine (IBM) 3590 tape cartridge is one example of a tape device that could be used as a backup or secondary storage system.

[0004] When the DASD fills to a predetermined threshold, the logical volume data for a selected logical volume is appended onto a tape cartridge, or a physical volume, with the original left on the DASD for possible cache hits. When a DASD file has been appended to a tape cartridge and the original remains on the DASD, the file is "premigrated."

[0005] When the host reads a logical volume from the VTS, a cache hit occurs if the logical volume currently resides on the DASD. If the logical volume is not on the DASD, a storage manager determines which of the physical tape volumes contains the logical volume. The corresponding physical volume is then mounted on one of the tape devices, and the data for the logical volume is transferred back to the DASD from the tape.

[0006] The logical volumes processed by a VTS fully emulate data volumes stored on physical storage media, such as tape media. Applications that utilize tape media for data storage typically employ a standard tape label at the beginning of the tape. The tape label is separated from the rest of the data files on the tape by a tapemark. To illustrate, the IBM standard tape label consists of three fields identified, for example, by VOL1, HDR1 and HDR2. Certain host operating systems, such as the IBM mainframe operating system type environment (MVS), whenever a tape volume is opened, the tape label is read. Data contained within the tape label, which had been written by the host operating system, identifies the data tape as well as the major dataset, i.e., data file, names contained on the data tape. The information in the tape label is then utilized by the host based tape operating system to verify the name and contents of the data tape.

[0007] The tape label, or header, is typically read by the host system, to ensure that the data tape is the correct data tape even if the data tape is being mounted as a "scratch" tape. If the entire content of a logical volume, i.e., emulated tape, has already been transferred to a data tape, a subsequent recall of the data on the data tape will be required to satisfy a read request for the header information. This, in turn, will require a mount of the physical data tape that contains the requested information, even if the logical volume is going to be overwritten, in which case a physical tape mount would not be necessary.

[0008] To minimize these unnecessary data tape mounts, whenever a logical volume is transferred, i.e., migrated, to a data tape, enough data to contain the entire tape label and its trailing tape mark is copied and stored in a "stub" on the DASD. Since header information for the data tapes are already present on the DASD, label reads for scratch mounts will not necessitate a physical tape mount for the particular data tapes. Generally, this information is not backed up as in the case of the data tapes. This, in turn, makes it difficult to quickly import and export data tapes to and from a VTS because all the data files must be read by the receiving VTS, in the case of an import operation, to reconstruct the stub information for the receiving VTS.

[0009] Accordingly, what is needed in the art is an improved method for importing/exporting data tapes from one VTS to another VTS that mitigates the above-discussed limitations in the prior art. More particularly, what is needed in the art is an improved method for importing/exporting data tapes that does not require reading every data file in the imported data tapes to generate the "stub" information on the receiving VTS's disk volume cache.

[0010] The present invention provides, in a first aspect, a method for efficiently importing/exporting a removable storage volume from a first virtual storage system to a second virtual storage system utilizing a plurality of data fragments, said method comprising the steps of: writing said plurality of data fragments to the end of said removable storage volume in said first virtual storage system, wherein each of said plurality of data fragments is associated with a corresponding data file located in said removable storage volume; transferring said removable storage volume to said second virtual storage system; and updating a tape volume cache in said second virtual storage system utilizing said plurality of data fragments without having to read each of said plurality of data files.

[0011] Preferably, each of said plurality of data fragments includes: at least one data file; and a file header preceding said at least one data file.

[0012] Preferably, said each of said plurality of data fragments further includes a data fragment trailer.

[0013] Preferably, said step of writing a data fragment to the end of a removable storage volume is initiated when said removable storage volume is closed.

[0014] Preferably, said step of writing said plurality of data fragments to the end of a removable storage volume is initiated after a period of time when said removable storage volume is idle.

[0015] Preferably, said plurality of data fragments are constructed utilizing information residing in a tape volume cache in said first virtual storage system.

[0016] Preferably, said first and second virtual storage systems are a virtual tape server.

[0017] The present invention provides, in a second aspect, a virtual storage system comprising: a direct access storage device; a plurality of removable storage volumes, operatively coupled to said direct access storage device to receive data; and a storage manager, coupled to said direct access storage device and said plurality of removable storage volumes, wherein said storage manager writes a plurality of data fragments to the end of a removable storage volume prior to exporting said removable storage volume, each of said plurality of data fragments is associated with a corresponding data file in said removable storage volume.

[0018] The present invention provides, in a third aspect, a virtual storage system, comprising: a direct access storage device; a plurality of removable storage volumes, operatively coupled to said direct access storage device to receive data; and a storage manager, coupled to said direct access storage device and said plurality of removable storage volumes, wherein said storage manager reads a plurality of data fragments of a removable storage volume having a plurality of data files received from a second virtual storage system and updates a tape volume cache utilizing information contained in said plurality of data fragments without having to read each of said plurality of data files in said removable storage volume.

[0019] Preferably, each of said plurality of data fragments includes: at least one data file; and a file header preceding said at least one data file.

[0020] Preferably, said each of said plurality of data fragments further includes a data fragment trailer.

[0021] Preferably, said plurality of data fragments is written to the end of said removable storage volume when said removable storage volume is closed.

[0022] Preferably, said plurality of data fragments is written to the end of said removable storage volume after a period of time when said removable storage volume is idle.

[0023] Preferably, said plurality of data fragments is constructed utilizing information residing in a tape volume cache in said virtual storage system.

[0024] Preferably, said removable storage volume is a data tape.

[0025] The present invention provides, in a fourth aspect, a method for formatting a data tape that allows for efficient importing/exporting of said data tape from a first virtual storage system to a second virtual storage system, said method comprising the steps of: writing a plurality of data files to said data tape; utilizing a plurality of filemarks to separate said plurality of data files from each other; and writing a plurality of data fragments at the end of said data tape, wherein each of said plurality of data fragments is associated with a data file in said data tape allowing said second virtual storage system to update a tape volume cache in said second virtual storage system without having to read any of said plurality of data files in said data tape.

[0026] The present invention provides, in a fifth aspect, a computer program product, comprising: a computer-readable medium having stored thereon computer executable instructions for implementing a method for efficiently importing/exporting a removable storage volume from a first virtual storage system to a second virtual storage system, said computer executable instructions when executed perform the steps of: writing a plurality of data fragments to the end of said removable storage volume in said first virtual storage system, wherein each of said plurality of data fragments is associated with a corresponding data file located in said removable storage volume; transferring said removable storage volume to said second virtual storage system; and updating a tape volume cache in said second virtual storage system utilizing said plurality of data fragments without having to read each of said plurality of data files.

[0027] Preferably, each of said data fragments includes: at least one data file; and a file header preceding said at least one data file.

[0028] Preferably, said each of said data fragments further includes a data fragment trailer.

[0029] Preferably, said step of writing a plurality of data fragments to the end of a removable storage volume is initiated when said removable storage volume is closed.

[0030] Preferably, said step of writing a plurality of data fragments to the end of a removable storage volume is initiated after a period of time when said removable storage volume is idle.

[0031] Preferably, said plurality of data fragments is constructed utilizing information residing in a tape volume cache in said first virtual storage system.

[0032] Preferably, said removable storage volume is a data tape.

[0033] The present invention accordingly provides an improved virtual storage system.

[0034] The present invention preferably provides a method and system utilizing data fragments for efficiently importing/exporting removable storage volumes between virtual storage systems.

[0035] The present invention preferably provides a method and system utilizing data fragments for efficiently importing/exporting a removable storage volume having a number of data files from a first virtual storage system to a second virtual storage system is disclosed. The method includes writing data fragments to the end of the

removable storage volume in the first virtual storage system. The data fragments contain information, such as data file headers, that uniquely identifies the data files residing in the removable storage volume. Next, the removable storage volume is transferred to the second virtual storage system. Upon receipt of the removable storage volume; the second virtual storage system updates a tape volume cache in the second virtual storage system utilizing the information contained in the data fragments without having to read each of the data files. In a related embodiment, the data fragments include at least one data file, a file header preceding the data file and a data fragment trailer. The data fragments are written to the end of the removable storage volume when the removable storage volume is closed.

[0036] In another embodiment of the present invention, the data fragments are written to the end of the removable storage volume after the removable storage volume has been idle for a period of time. This has the added advantage that if an intervening system crash occurs or a tape volume cache is damaged before the removable storage volume is filled, the data fragments in the partially filled removable storage volume can be utilized to restore the tape volume cache.

[0037] A preferred embodiment of the present Invention will now be described, by way of example only, with reference to the accompanying drawings, in which:

FIGURE 1 illustrates a block diagram of an exemplary virtual storage system that provides a suitable environment for the practice of a preferred embodiment of the present invention;

FIGUREs 2A and 2B illustrate a simplified representation of an embodiment of an organization of contents in a removable storage volume according to a preferred embodiment of the present invention;

FIGURE 3 illustrates a high-level process flow diagram for writing to a removable storage volume utilizing the principles disclosed by a preferred embodiment of the present invention; and

FIGURE 4 illustrates a process flow diagram for importing a data tape into a virtual storage system from another virtual storage system according to a preferred embodiment of the present Invention.

[0038] With reference now to the figures and in particular with reference to FIGURE 1, there is depicted a block diagram of an exemplary virtual storage system 100 that provides a suitable environment for the practice of a preferred embodiment of the present invention. Virtual storage system 100 includes a virtual tape server 110, a tape library 150 and a library manager 145. A host system 10 is linked to virtual tape server 110 via a network connection, e.g. TCP/IP, LAN, Ethernet, the IBM Enterprise System Connection (ESCON). In a preferred embodiment, host system 10 is a computer, such as a personal computer, workstation or mainframe, that is linked to the virtual tape server 110 via an ESCON channel. Virtual tape server 110, in an advantageous embodiment, is a computer, such as a personal computer, workstation or mainframe and is associated with a Direct Access Storage Device (DASD) cache 135. In a preferred embodiment, DASD cache 135 includes a plurality of hard disks that are spaced into redundant array of inexpensive disk (RAID) arrays. A separate data cache, i.e., tape volume cache 135a, is also shown in FIGURE 1 and, in other advantageous embodiments, may be part of DASD 135.

[0039] Tape library 150 includes a plurality of tape drives, generally designated first, second and third tape drives 155a, 155b, 155c, such as the International Business Machine (IBM) Magstar 3590 tape drives. Generally, a removable storage volume, i.e., a tape cartridge, is loaded into each of the tape drives. Tape library 150, typically includes storage management software utilized to monitor the active space on the tape cartridges and schedule reclamations of tape cartridges when the system is less active. In an advantageous embodiment, tape library 150 is a tape library system such as the IBM Magstar 3494 Tape Library. Library manager 145 is utilized in virtual storage system 100 to install, maintain, configure, and operate tape library 150. Library manager 145 includes a controller (not shown), such as a personal computer or workstation that can assume direct control over tape library 150.

[0040] DASD cache 135 provides a cache for data stored in tape library 150. Tape volume cache 135a maintains logical volumes as logical volume files that are concatenated into physical volume files in the tape cartridges loaded in the tape drives located within tape library 150. When a logical volume file in DASD cache 135 moves to a tape drive in tape library 150, the logical volume file is written to a physical volume file on a tape cartridge in the actual tape drive. When a physical volume file is recalled from a tape drive and moved to tape volume cache 135a, the physical volume file then becomes a logical volume file in the tape volume cache 135a. In this way, tape volume cache 135a provides a window to host system 10 of all the physical volumes files in tape library 150.

[0041] Upon initialization, virtual tape server 110 loads a virtual tape controller 115 into random access memory (RAM). Virtual tape controller 115 includes a plurality of virtual tape daemons, generally designated first and second tape daemons 120a, 120b (for ease of explanation) that represent and emulate virtual tape devices to host system 10. Host system's 10 operating system, in turn, manages the presentation of the virtual tape devices to system users (not shown). Host system 10 views the virtual tape devices as actual tape drives and when host system 10 attempts to access a logical volume in a selected virtual tape device, the respective virtual tape daemon associated with the virtual tape device

requested by the host system 10 will handle the host access request.

[0042] A hierarchical storage management (HSM) client program 125 within virtual tape server 110 intercepts- and processes the access request from the virtual tape daemons. HSM client 125 then carries out host system 10 request to access the logical volume file in tape volume cache 135a. In a preferred embodiment, HSM client program 125 is part of the IBM ADSTAR Distributed Storage Manager (ADSM) product. The ADSM provides generic client/server HSM functions and includes an ADSM client to handle file access requests with software integrated with the operating system kernel.

[0043] Virtual tape server 110 also includes a storage manager server 140 that handles data transfers between DASD cache 135 and tape library 150. For example, if HSM client 135 attempts to mount a logical volume file that is not located in tape volume cache 135a, HSM client 125 will communicate the access request to storage manager server 140. If the tape in the access request is already mounted in a tape drive in tape library 150, storage manager server 140 will access the physical volume for the requested logical volume file from the mounted tape. However, if the requested file on a tape is not presently mounted in a tape drive, storage manager server 140 will initiate a request to library manager 145 to mount the tape containing the physical volume corresponding to the requested logical volume file. In preferred embodiments, the storage manager server 140 is part of the IBM ADSM product described above.

[0044] In a preferred embodiment, storage manager server 140 migrates entire logical volume files from tape volume cache 135a to tape library 150. When the available space in tape volume cache 135a reaches a predetermined level or after a predetermined period of time, an automated systems administrator 130 will direct storage manager server 140 to migrate logical volume files from tape volume cache 135a to tape library 150 for archival therein. HSM client 125 will then substitute the migrated logical volume file with a stub file that includes all the information needed to locate and recall a physical volume file from tape library 150 that corresponds to the logical volume. However, when HSM client 125 attempts to access information not included in a stub file, HSM client 125 will request that storage manager server 140 recall the logical volume file from the physical volume in tape library 150 to replace the stub file in tape volume cache 135a. Typically, HSM client 125 would migrate the least used logical volume files.

[0045] Automated systems administrator 130 is included in virtual tape server 110 to perform operations that typically performed by a human system administrator. Automated system administrator 130 filters any error messages concerning tape library 150 that are generated by storage manager server 140 that, in turn, receives error information updates from library manager 145. Typically, automated system administrator 130 stores information associated with the physical volumes in an associated volume status table (not shown). Automated system administrator 130 also utilizes a premigration table (not shown) that maintains information on any logical volume files that are in the process of being premigrated from tape volume cache 135a to tape library 150. During premigration, a logical volume file is locked while storage manager server 140 copies from the logical volume file in tape volume cache 135a to tape library 150. The file name is placed in the premigration table to indicate that the logical volume file is presently in the process of being premigrated. Once the logical volume file is copied over to tape library 150, the logical volume file name is removed from the premigration table. At this point, the logical volume file is maintained in both tape volume cache 135a and tape library 150. When the available space in tape volume cache 135a reaches a predetermined low threshold, the logical volume files that have been premigrated, such that a copy is maintained in both tape volume cache 135a and tape library 150, are deleted from tape volume cache 135a and replaced with a stub file. The process of deleting the logical volume files from tape volume cache 135a and replacing the moved files with a stub file is referred to herein as migration.

[0046] Whenever virtual tape server 110 processes a data tape for a scratch allocation operation, virtual tape server 110 first performs validation steps to ensure that the data tape mounted is usable as a scratch volume. The validation steps includes reading information from the data tape's volume label, or header. The information contained in the volume header typically contains the volume name, ownership and first data file name on the data tape. For example, the IBM 3494 Model B16 Virtual Tape Server system stores the first few records of all the logical volumes managed by the virtual tape server in tape volume cache 135a. This is accomplished even if the rest of the contents of a logical volume has been migrated out to a physical data tape. This allows a scratch mount request to be satisfied without having to recall the image of the logical volume from its corresponding physical data tape. These records, contained in a data fragment and stored in tape volume cache 135a are utilized by host system 10 to accomplish the validation that a volume is eligible for use as a scratch volume. One example of the construction and contents of data fragments is disclosed in U.S. Patent Application Serial No. 08/919,043, entitled "Storage and Access to Scratch Mounts in VTS system," filed August 27, 1997. Since typically 50% of data tape mounts are scratch mounts, the utilization of these data fragments enhances the overall system performance of virtual tape server 110.

[0047] Referring now to FIGURES 2A and 2B, there are illustrated simplified representations of an embodiment of an organization of contents 200 in a removable storage volume according to a preferred embodiment of the present invention. The formatting of the contents

200 in the removable storage volume, such as magnetic tape, begins with a tape label 205 that uniquely identifies this particular removable storage volume, e.g., the volume serial number. A first tapemark 210a, e.g., a unique sequence of bits, separates tape label 205 from first logical volume data 215 that may contain data files corresponding to a particular customer or project. As shown in FIGURE 2A, the removable storage volume also includes second logical volume data 220 up to N logical volumes data 230 followed by a second tapemark 210b, also known as a "filemark," to indicate the end of the logical volume data. Generally, data blocks, e.g., tape label 205 and first logical volume data, stored on a removable storage volume are organized in groups forming structures of two kinds: user-defined data sets or "data files" and "labels" that are a group of blocks that identify and describe the removable storage volume and/or a data file. The next contiguous block in the removable storage volume is a data fragment 240 file that, in an advantageous embodiment, is a data file. A third tapemark 210c separates data fragments 240 from an end of data mark 250, i.e., end of volume indication, that may, for example, be two contiguous tapemarks.

[0048] Data fragments 240, as depicted in FIGURE 2B, contains fragment of data from each of the data files residing in the removable storage volume, such as header information. Generally, the data fragment includes ADSM information utilized in recalling the data file, VTS header information that identifies the logical volume and may provide, in an advantageous embodiment, size and timestamp information. The data fragment may also contain, for example, the first 3 Kbytes of a customer data as represented internally by a VTS. To illustrate, the customer data may begin with VOL1, HDR1, HDR2 and a tape mark. The VOL and HDR records are typically 80 bytes each and including the VTS format overhead, will fit in a data fragment easily. It is this "standard label" information that a host system will typically read on a scratch mount operation to verify that it has the right tape. Utilization of a scratch tape will then usually continue with the host system re-writing the tape from the tape's beginning, at which point the data in the logical volume (and data fragment) will no longer be required, having been replaced by the write data.

[0049] In a preferred embodiment, data fragments 240 is written whenever a removable storage volume is closed and the information (discussed above) in data fragments 240 is generated from information that is already residing in a tape volume cache. The virtual storage system's tape volume cache may be resident in a tape volume cache associated with the virtual storage system or, alternatively, may be resident in a separate direct access storage device. Thus, when the removable storage volume is exported, i.e., transferred to another virtual storage system, the information in data fragments 240 can be read quickly and more efficiently by the receiving virtual storage system to reconstruct the tape volume cache information in the receiving virtual storage system tape volume cache without having to read the entire content of the transferred removable storage volume or scan through the entire database if data fragments 240 was appended to the transferring virtual storage system database in an alternate embodiment. This results in faster operational readiness and enhanc s overall system performance.

[0050] Alternatively, in another advantageous embodiment, data fragments 240 is written to the stacked, i.e., mounted, removable storage volume whenever the stacked removable storage volume has been idle, i.e., not used, for a specified period of time. Thus, if an intervening system crash occurs before the removable storage volume is filled, data fragments will still exist, even for a partially filled storage volume. The data fragments can then be used to update the virtual storage system tape volume cache during the subsequent recovery operations. Utilizing the information in the data fragments to restore the crashed virtual storage system, tape volume cache instead of having to read each data file in the associated removable storage volume to obtain the necessary information results in faster recovery times and enhances system performance. The process of generating of data fragments and the utilization of the data fragments in an import operation is described in greater detail hereinafter with respect to FIGURES 3 and 4, with continuing reference to FIGURES 1 and 2.

[0051] Referring now to FIGURE 3, there is depicted a high- level process flow diagram 300 for writing to a removable storage volume utilizing the principles disclosed by a preferred embodiment of the present invention. Process flow diagram 300 is initiated, as illustrated in step 310, when one or more logical volumes in tape volume cache 135a is scheduled to be copied onto a data tape, i.e., removable storage volume, in tape library 150. The copying of a logical volume to a physical data tape may be necessitated if data in tape volume cache 135a has reached a predetermined level. Storage manager server 140 will then select a data tape that is not full as depicted in step 315. After a non-full data tape has been selected, library manager 145 will execute the command to mount the data tape on one of the tape drives in tape library 150, following which, virtual tape server 110 will open the data tape, as illustrated in step 320. Following the mounting of the selected data tape on a tape drive, as depicted in step 325, storage manager server 140 positions a read/write head in the tape drive to the end of the data in the data tape and backspaces one data file. The data tape is now positioned prior to the data tape's data fragments file. Alternatively, in another advantageous embodiment, virtual tape server 110 may start at the beginning of the data tape and forward space to second tapemark 210b. Next, storage manager server 140 will migrate, i.e., copy, a logical volume file to the data tape.

[0052] After the logical volume file has been written onto the data tape, process 300 determines if there is additional space available on the opened data tape for

more logical volume files, as illustrated in decisional step 335. If process 300 determines if there is no more space available on the opened data tape, the opened data tape is marked as full, as depicted in step 340. After the opened data tape is marked as full, the data fragment file in the opened data tape is updated, i.e., a data fragment of the newly migrated logical volume is added to the existing data fragments, and the opened data tape is closed and demounted, i.e., archived, as illustrated in step 365.

[0053] Returning back to decisional step 335, if process 300 determines that there is more space available on the opened data tape for more data files, another determination is made, as depicted in decisional step 345, to ascertain if there are more logical volumes that need to be copied onto the opened data tape. If there is another logical volume that needs to be copied, process 300 proceeds to step 300 to initiate another copying operation. However, if process 300 determines that there are no more logical volumes queued for migration, process 300 resorts to an idle state for a predetermined period of time, as illustrated in step 350. After waiting for the specified period of time, process 300 again checks to see if another logical volume is scheduled to be migrated from tape volume cache 135a to the opened data tape, as depicted in decisional step 335. If there is another logical volume scheduled to be migrated to the opened data tape, process 300 proceeds back to step 330 to initiate another copying operation. If, on the other hand, process 300 determines that there is no logical volume in tape volume cache 135a required to be migrated to the opened data tape, process 300 makes another determination, as illustrated in decisional step 360, to ascertain if the opened data tape has been idled for a predetermined period of time.

[0054] The predetermined period of time may, in an advantageous embodiment, be a set number of wait cycles. It should be noted that a preferred embodiment of the present invention does not contemplate limiting its practice to any one set period of idle time or a specific number of wait cycles. If process 300 determines that the opened data tape has not been idled for the specified period of time, process 300 proceeds back to step 350, where process 300 reverts back to an idle state for the specified period of time. If, however, process 300 determines that the opened data tape has been idled for a period equal to or greater than the predetermined waiting period, process 300 proceeds on to step 365 discussed previously, where the associated data fragment file in the opened data tape is updated with the data fragments that uniquely identifies all the newly copied logical volumes. After the data fragments have been updated, the opened data tape is then closed and demounted and process 300 is terminated, as depicted in step 370.

[0055] Referring now to FIGURE 4, there is illustrated a process flow diagram 400 for importing a data tape into a virtual storage system from another virtual storage system according to a preferred embodiment of the present invention. Process 400 is initiated, as illustrated in step 410, when a data tape is transferred from one virtual storage system to another. The receiving virtual storage system mounts and opens the data tape utilizing one its available tape drives in its tape library, as depicted in step 420. The receiving virtual storage system then opens and positions th data tape to its end of the data field and backspaces one data file. Alternatively, in another advantageous embodiment, the receiving virtual tape server may start at the beginning of the data tape and forward space to the data fragment file. The opened data tape is now positioned at the beginning of the opened data tape associated data fragment file, as illustrated in step 430. Process 400 then reads the data fragments and copies the information that uniquely identifies the data files recorded in the data tape to a tape volume cache associated with the receiving virtual storage system, as depicted in step 440.

[0056] Alternatively, in another advantageous embodiment, process 400 may also be used to more efficiently restore a virtual storage system's tape volume cache following, for example, a system crash that had corrupted or erase the contents in the tape volume cache. In this alternate embodiment, following a system crash, each of the data tapes in an associated tape library is mounted and opened. The data fragments in each data tape is then read and copied to the virtual storage system's tape volume cache to restore the information that had been lost or corrupted. Having to only read a single data file, i.e., data fragment file, in each data tape instead of having read all the data files in each data tape to recover the information lost in the tape volume cache results in a shorter recovery operation from a system crash that ultimately enhances overall system performance and reliability.

[0057] It should be noted that although a preferred embodiment of the present invention has been described, in one embodiment, in the context of a computer system, those skilled in the art will readily appreciate that the methods of a preferred embodiment of the present invention described hereinabove may be implemented, for example, by operating storage manager server 140 or other suitable electronic module to execute a corresponding sequence of machine-readable instructions. These instructions may reside in various types of signal- bearing media. In this respect, one aspect of a preferred embodiment of the present invention concerns a programmed product, that includes signal-bearing media tangibly embodying a program of machine-readable instructions executable by a digital data processor to perform the methods described above. A preferred embodiment of the present invention does not contemplate limiting its practice to any particular type of signal- bearing media, i.e., computer readable medium, utilized to actually carry out the distribution. Examples of signal-bearing media includes recordable type media, such as floppy disks and hard disk drives, and transmission type media such as digital and analog communica-

tion links and wireless.

Claims

1. A method for efficiently importing/exporting a removable storage volume from a first virtual storage system to a second virtual storage system utilizing a plurality of data fragments, said method comprising the steps of:

   writing said plurality of data fragments to the end of said removable storage volume in said first virtual storage system, wherein each of said plurality of data fragments is associated with a corresponding data file located in said removable storage volume;

   transferring said removable storage volume to said second virtual storage system; and

   updating a tape volume cache in said second virtual storage system utilizing said plurality of data fragments without having to read each of said plurality of data files.

2. A virtual storage system comprising:

   a direct access storage device;

   a plurality of removable storage volumes, operatively coupled to said direct access storage device to receive data; and

   a storage manager, coupled to said direct access storage device and said plurality of removable storage volumes,

   wherein said storage manager writes a plurality of data fragments to the end of a removable storage volume prior to exporting said removable storage volume, each of said plurality of data fragments is associated with a corresponding data file in said removable storage volume.

3. A virtual storage system as claimed in claim 2, wherein said storage manager reads a plurality of data fragments of a removable storage volume having a plurality of data files received from a second virtual storage system and updates a tape volume cache utilizing information contained in said plurality of data fragments without having to read each of said plurality of data files in said removable storage volume.

4. A method as claimed in claim 1, or a system as claimed claim 2 or claim 3, wherein each of said plurality of data fragments includes:

   at least one data file; and

   a file header preceding said at least one data file.

5. A method or system as claimed in claim 4, wherein said each of said plurality of data fragments further includes a data fragment trailer.

6. A method or system as claimed in any preceding claim, wherein said plurality of data fragments is written to the end of said removable storage volume when said removable storage volume is closed.

7. A method or system as claimed in any preceding claim, wherein said plurality of data fragments is written to the end of said removable storage volume after a period of time when said removable storage volume is idle.

8. A method or system as claimed in any preceding claim,, wherein said plurality of data fragments is constructed utilizing information residing in a tape volume cache in said virtual storage system.

9. A method or system as claimed in any preceding claim, wherein said removable storage volume is a data tape.

10. A computer program comprising computer program instructions to, when loaded into a computer system and executed, cause the computer system to perform the steps of a method as claimed in claim 1 or in any of claims 4 to 7.

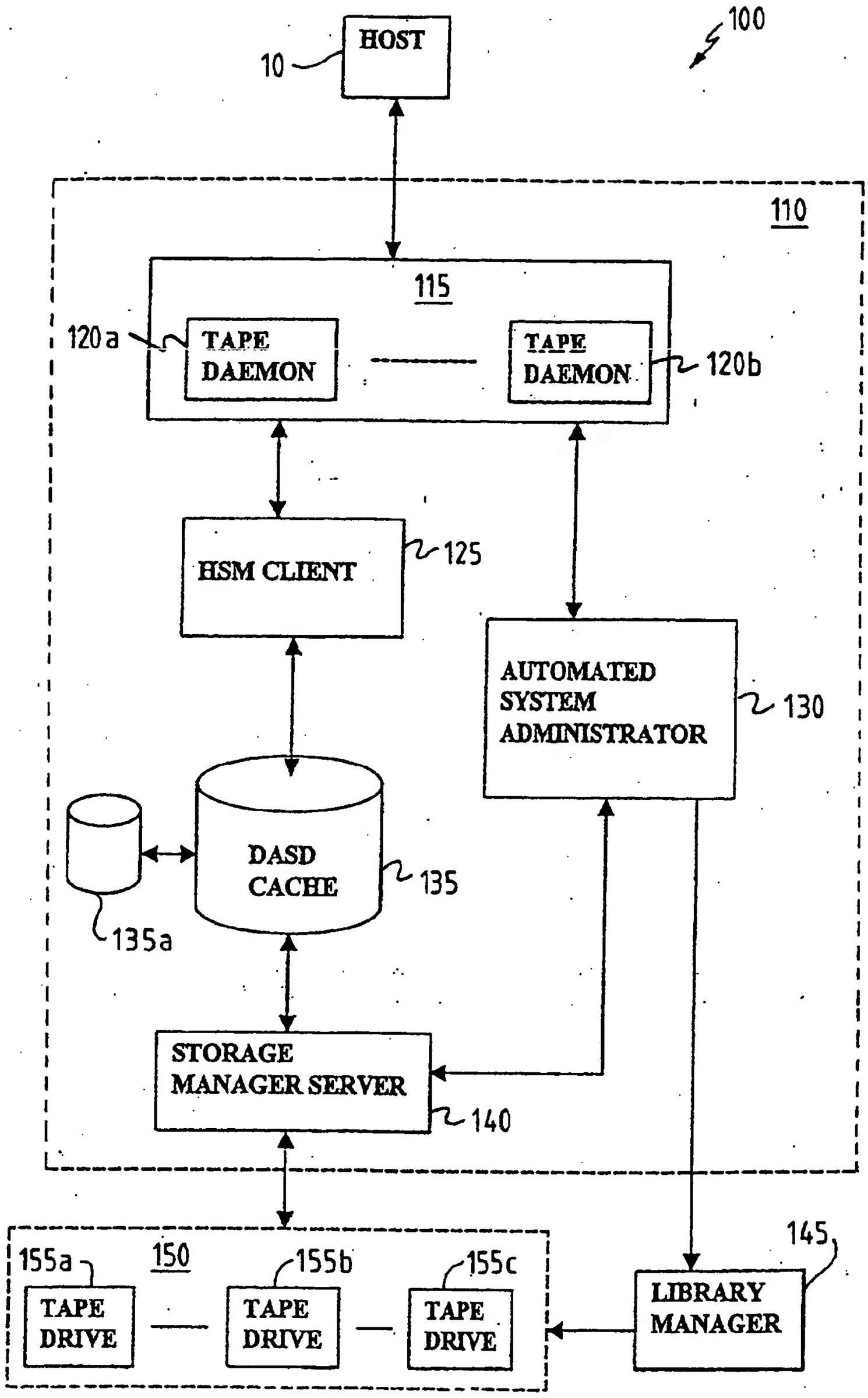

FIG. 1 PRIOR ART

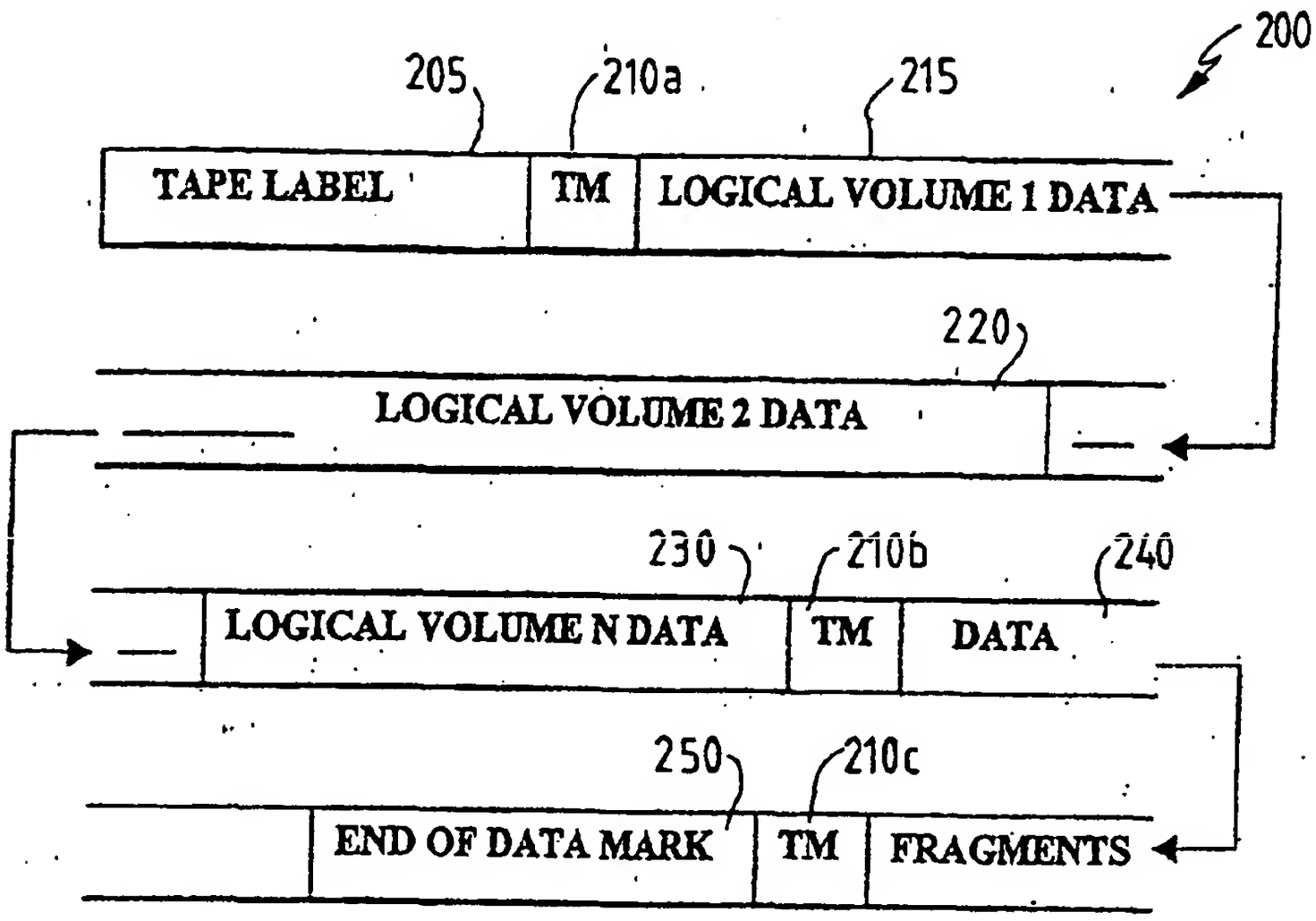
200
205
210a
215
TAPE LABEL
TM
LOGICAL VOLUME 1 DATA
220
LOGICAL VOLUME 2 DATA
230
210b
240
LOGICAL VOLUME N DATA
TM
DATA
250
210c
END OF DATA MARK
TM
FRAGMENTS

FIG. 2A

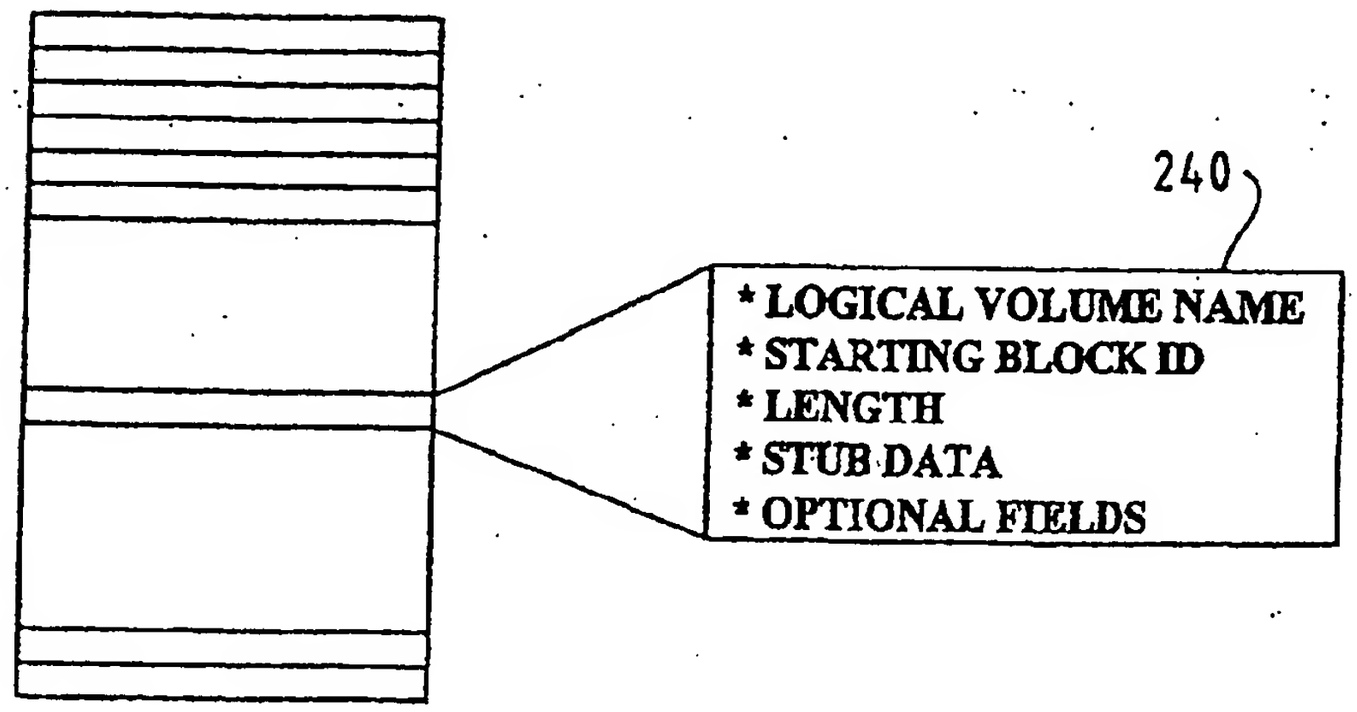
240
* LOGICAL VOLUME NAME
* STARTING BLOCK ID
* LENGTH
* STUB DATA
* OPTIONAL FIELDS

FIG. 2B